

Exam Questions AWS-Certified-Data-Engineer-Associate

AWS Certified Data Engineer - Associate (DEA-C01)

<https://www.2passeasy.com/dumps/AWS-Certified-Data-Engineer-Associate/>



NEW QUESTION 1

A company uses Amazon Athena for one-time queries against data that is in Amazon S3. The company has several use cases. The company must implement permission controls to separate query processes and access to query history among users, teams, and applications that are in the same AWS account. Which solution will meet these requirements?

- A. Create an S3 bucket for each use case
- B. Create an S3 bucket policy that grants permissions to appropriate individual IAM user
- C. Apply the S3 bucket policy to the S3 bucket.
- D. Create an Athena workgroup for each use case
- E. Apply tags to the workgroup
- F. Create an IAM policy that uses the tags to apply appropriate permissions to the workgroup.
- G. Create an IAM role for each use case
- H. Assign appropriate permissions to the role for each use case
- I. Associate the role with Athena.
- J. Create an AWS Glue Data Catalog resource policy that grants permissions to appropriate individual IAM users for each use case
- K. Apply the resource policy to the specific tables that Athena uses.

Answer: B

Explanation:

Athena workgroups are a way to isolate query execution and query history among users, teams, and applications that share the same AWS account. By creating a workgroup for each use case, the company can control the access and actions on the workgroup resource using resource-level IAM permissions or identity-based IAM policies. The company can also use tags to organize and identify the workgroups, and use them as conditions in the IAM policies to grant or deny permissions to the workgroup. This solution meets the requirements of separating query processes and access to query history among users, teams, and applications that are in the same AWS account. References:

? Athena Workgroups

? IAM policies for accessing workgroups

? Workgroup example policies

NEW QUESTION 2

A company stores datasets in JSON format and .csv format in an Amazon S3 bucket. The company has Amazon RDS for Microsoft SQL Server databases, Amazon DynamoDB tables that are in provisioned capacity mode, and an Amazon Redshift cluster. A data engineering team must develop a solution that will give data scientists the ability to query all data sources by using syntax similar to SQL.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Use AWS Glue to crawl the data source
- B. Store metadata in the AWS Glue Data Catalog
- C. Use Amazon Athena to query the data
- D. Use SQL for structured data source
- E. Use PartiQL for data that is stored in JSON format.
- F. Use AWS Glue to crawl the data source
- G. Store metadata in the AWS Glue Data Catalog
- H. Use Redshift Spectrum to query the data
- I. Use SQL for structured data source
- J. Use PartiQL for data that is stored in JSON format.
- K. Use AWS Glue to crawl the data source
- L. Store metadata in the AWS Glue Data Catalog
- M. Use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv format
- N. Store the transformed data in an S3 bucket
- O. Use Amazon Athena to query the original and transformed data from the S3 bucket.
- P. Use AWS Lake Formation to create a data lake
- Q. Use Lake Formation jobs to transform the data from all data sources to Apache Parquet format
- R. Store the transformed data in an S3 bucket
- S. Use Amazon Athena or Redshift Spectrum to query the data.

Answer: A

Explanation:

The best solution to meet the requirements of giving data scientists the ability to query all data sources by using syntax similar to SQL with the least operational overhead is to use AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, use Amazon Athena to query the data, use SQL for structured data sources, and use PartiQL for data that is stored in JSON format.

AWS Glue is a serverless data integration service that makes it easy to prepare, clean, enrich, and move data between data stores¹. AWS Glue crawlers are processes that connect to a data store, progress through a prioritized list of classifiers to determine the schema for your data, and then create metadata tables in the Data Catalog². The Data Catalog is a persistent metadata store that contains table definitions, job definitions, and other control information to help you manage your AWS Glue components³. You can use AWS Glue to crawl the data sources, such as Amazon S3, Amazon RDS for Microsoft SQL Server, and Amazon DynamoDB, and store the metadata in the Data Catalog.

Amazon Athena is a serverless, interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL or Python⁴. Amazon Athena also supports PartiQL, a SQL-compatible query language that lets you query, insert, update, and delete data from semi-structured and nested data, such as JSON. You can use Amazon Athena to query the data from the Data Catalog using SQL for structured data sources, such as .csv files and relational databases, and PartiQL for data that is stored in JSON format. You can also use Athena to query data from other data sources, such as Amazon Redshift, using federated queries.

Using AWS Glue and Amazon Athena to query all data sources by using syntax similar to SQL is the least operational overhead solution, as you do not need to provision, manage, or scale any infrastructure, and you pay only for the resources you use. AWS Glue charges you based on the compute time and the data processed by your crawlers and ETL jobs¹. Amazon Athena charges you based on the amount of data scanned by your queries. You can also reduce the cost and improve the performance of your queries by using compression, partitioning, and columnar formats for your data in Amazon S3.

Option B is not the best solution, as using AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, and use Redshift Spectrum to query the data, would incur more costs and complexity than using Amazon Athena. Redshift Spectrum is a feature of Amazon Redshift, a fully managed data warehouse service, that allows you to query and join data across your data warehouse and your data lake using standard SQL. While Redshift Spectrum is powerful and useful for many data warehousing scenarios, it is not necessary or cost-effective for querying all data sources by using syntax similar to SQL. Redshift Spectrum charges you based on the amount of data scanned by your queries, which is similar to Amazon Athena, but it also requires you to have an

Amazon Redshift cluster, which charges you based on the node type, the number of nodes, and the duration of the cluster⁵. These costs can add up quickly, especially if you have large volumes of data and complex queries. Moreover, using Redshift Spectrum would introduce additional latency and complexity, as you would have to provision and manage the cluster, and create an external schema and database for the data in the Data Catalog, instead of querying it directly from Amazon Athena.

Option C is not the best solution, as using AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv format, store the transformed data in an S3 bucket, and use Amazon Athena to query the original and transformed data from the S3 bucket, would incur more costs and complexity than using Amazon Athena with PartiQL. AWS Glue jobs are ETL scripts that you can write in Python or Scala to transform your data and load it to your target data store. Apache Parquet is a columnar storage format that can improve the performance of analytical queries by reducing the amount of data that needs to be scanned and providing efficient compression and encoding schemes⁶. While using AWS Glue jobs and Parquet can improve the performance and reduce the cost of your queries, they would also increase the complexity and the operational overhead of the data pipeline, as you would have to write, run, and monitor the ETL jobs, and store the transformed data in a separate location in Amazon S3. Moreover, using AWS Glue jobs and Parquet would introduce additional latency, as you would have to wait for the ETL jobs to finish before querying the transformed data.

Option D is not the best solution, as using AWS Lake Formation to create a data lake, use Lake Formation jobs to transform the data from all data sources to Apache Parquet format, store the transformed data in an S3 bucket, and use Amazon Athena or Redshift Spectrum to query the data, would incur more costs and complexity than using Amazon Athena with PartiQL. AWS Lake Formation is a service that helps you centrally govern, secure, and globally share data for analytics and machine learning⁷. Lake Formation jobs are ETL jobs that you can create and run using the Lake Formation console or API. While using Lake Formation and Parquet can improve the performance and reduce the cost of your queries, they would also increase the complexity and the operational overhead of the data pipeline, as you would have to create, run, and monitor the Lake Formation jobs, and store the transformed data in a separate location in Amazon S3. Moreover, using Lake Formation and Parquet would introduce additional latency, as you would have to wait for the Lake Formation jobs to finish before querying the transformed data. Furthermore, using Redshift Spectrum to query the data would also incur the same costs and complexity as mentioned in option B. References:

- ? What is Amazon Athena?
- ? Data Catalog and crawlers in AWS Glue
- ? AWS Glue Data Catalog
- ? Columnar Storage Formats
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
- ? AWS Glue Schema Registry
- ? What is AWS Glue?
- ? Amazon Redshift Serverless
- ? Amazon Redshift provisioned clusters
- ? [Querying external data using Amazon Redshift Spectrum]
- ? [Using stored procedures in Amazon Redshift]
- ? [What is AWS Lambda?]
- ? [PartiQL for Amazon Athena]
- ? [Federated queries in Amazon Athena]
- ? [Amazon Athena pricing]
- ? [Top 10 performance tuning tips for Amazon Athena]
- ? [AWS Glue ETL jobs]
- ? [AWS Lake Formation jobs]

NEW QUESTION 3

A company stores petabytes of data in thousands of Amazon S3 buckets in the S3 Standard storage class. The data supports analytics workloads that have unpredictable and variable data access patterns.

The company does not access some data for months. However, the company must be able to retrieve all data within milliseconds. The company needs to optimize S3 storage costs.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Use S3 Storage Lens standard metrics to determine when to move objects to more cost- optimized storage classe
- B. Create S3 Lifecycle policies for the S3 buckets to move objects to cost-optimized storage classe
- C. Continue to refine the S3 Lifecycle policies in the future to optimize storage costs.
- D. Use S3 Storage Lens activity metrics to identify S3 buckets that the company accesses infrequentl
- E. Configure S3 Lifecycle rules to move objects from S3 Standard to the S3 Standard-Infrequent Access (S3 Standard-IA) and S3 Glacier storage classes based on the age of the data.
- F. Use S3 Intelligent-Tierin
- G. Activate the Deep Archive Access tier.
- H. Use S3 Intelligent-Tierin
- I. Use the default access tier.

Answer: D

Explanation:

S3 Intelligent-Tiering is a storage class that automatically moves objects between four access tiers based on the changing access patterns. The default access tier consists of two tiers: Frequent Access and Infrequent Access. Objects in the Frequent Access tier have the same performance and availability as S3 Standard, while objects in the Infrequent Access tier have the same performance and availability as S3 Standard-IA. S3 Intelligent-Tiering monitors the access patterns of each object and moves them between the tiers accordingly, without any operational overhead or retrieval fees. This solution can optimize S3 storage costs for data with unpredictable and variable access patterns, while ensuring millisecond latency for data retrieval. The other solutions are not optimal or relevant for this requirement. Using S3 Storage Lens standard metrics and activity metrics can provide insights into the storage usage and access patterns, but they do not automate the data movement between storage classes. Creating S3 Lifecycle policies for the S3 buckets can move objects to more cost-optimized storage classes, but they require manual configuration and maintenance, and they may incur retrieval fees for data that is accessed unexpectedly. Activating the Deep Archive Access tier for S3 Intelligent-Tiering can further reduce the storage costs for data that is rarely accessed, but it also increases the retrieval time to 12 hours, which does not meet the requirement of millisecond latency. References:

- ? S3 Intelligent-Tiering
- ? S3 Storage Lens
- ? S3 Lifecycle policies
- ? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

NEW QUESTION 4

A data engineer is configuring Amazon SageMaker Studio to use AWS Glue interactive sessions to prepare data for machine learning (ML) models. The data engineer receives an access denied error when the data engineer tries to prepare the data by using SageMaker Studio.

Which change should the engineer make to gain access to SageMaker Studio?

- A. Add the AWSGlueServiceRole managed policy to the data engineer's IAM user.
- B. Add a policy to the data engineer's IAM user that includes the sts:AssumeRole action for the AWS Glue and SageMaker service principals in the trust policy.
- C. Add the AmazonSageMakerFullAccess managed policy to the data engineer's IAM user.
- D. Add a policy to the data engineer's IAM user that allows the sts:AddAssociation action for the AWS Glue and SageMaker service principals in the trust policy.

Answer: B

Explanation:

This solution meets the requirement of gaining access to SageMaker Studio to use AWS Glue interactive sessions. AWS Glue interactive sessions are a way to use AWS Glue DataBrew and AWS Glue Data Catalog from within SageMaker Studio. To use AWS Glue interactive sessions, the data engineer's IAM user needs to have permissions to assume the AWS Glue service role and the SageMaker execution role. By adding a policy to the data engineer's IAM user that includes the sts:AssumeRole action for the AWS Glue and SageMaker service principals in the trust policy, the data engineer can grant these permissions and avoid the access denied error. The other options are not sufficient or necessary to resolve the error. References:

? Get started with data integration from Amazon S3 to Amazon Redshift using AWS Glue interactive sessions

? Troubleshoot Errors - Amazon SageMaker

? AccessDeniedException on sagemaker:CreateDomain in AWS SageMaker Studio, despite having SageMakerFullAccess

NEW QUESTION 5

A data engineer needs to schedule a workflow that runs a set of AWS Glue jobs every day. The data engineer does not require the Glue jobs to run or finish at a specific time.

Which solution will run the Glue jobs in the MOST cost-effective way?

- A. Choose the FLEX execution class in the Glue job properties.
- B. Use the Spot Instance type in Glue job properties.
- C. Choose the STANDARD execution class in the Glue job properties.
- D. Choose the latest version in the GlueVersion field in the Glue job properties.

Answer: A

Explanation:

The FLEX execution class allows you to run AWS Glue jobs on spare compute capacity instead of dedicated hardware. This can reduce the cost of running non-urgent or non-time sensitive data integration workloads, such as testing and one-time data loads. The FLEX execution class is available for AWS Glue 3.0 Spark jobs. The other options are not as cost-effective as FLEX, because they either use dedicated resources (STANDARD) or do not affect the cost at all (Spot Instance type and GlueVersion). References:

? Introducing AWS Glue Flex jobs: Cost savings on ETL workloads

? Serverless Data Integration – AWS Glue Pricing

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 125)

NEW QUESTION 6

A data engineer must manage the ingestion of real-time streaming data into AWS. The data engineer wants to perform real-time analytics on the incoming streaming data by using time-based aggregations over a window of up to 30 minutes. The data engineer needs a solution that is highly fault tolerant.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Use an AWS Lambda function that includes both the business and the analytics logic to perform time-based aggregations over a window of up to 30 minutes for the data in Amazon Kinesis Data Streams.
- B. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to analyze the data that might occasionally contain duplicates by using multiple types of aggregations.
- C. Use an AWS Lambda function that includes both the business and the analytics logic to perform aggregations for a tumbling window of up to 30 minutes, based on the event timestamp.
- D. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to analyze the data by using multiple types of aggregations to perform time-based analytics over a window of up to 30 minutes.

Answer: A

Explanation:

This solution meets the requirements of managing the ingestion of real-time streaming data into AWS and performing real-time analytics on the incoming streaming data with the least operational overhead. Amazon Managed Service for Apache Flink is a fully managed service that allows you to run Apache Flink applications without having to manage any infrastructure or clusters. Apache Flink is a framework for stateful stream processing that supports various types of aggregations, such as tumbling, sliding, and session windows, over streaming data. By using Amazon Managed Service for Apache Flink, you can easily connect to Amazon Kinesis Data Streams as the source and sink of your streaming data, and perform time-based analytics over a window of up to 30 minutes. This solution is also highly fault tolerant, as Amazon Managed Service for Apache Flink automatically scales, monitors, and restarts your Flink applications in case of failures. References:

? Amazon Managed Service for Apache Flink

? Apache Flink

? Window Aggregations in Flink

NEW QUESTION 7

A company created an extract, transform, and load (ETL) data pipeline in AWS Glue. A data engineer must crawl a table that is in Microsoft SQL Server. The data engineer needs to extract, transform, and load the output of the crawl to an Amazon S3 bucket. The data engineer also must orchestrate the data pipeline.

Which AWS service or feature will meet these requirements MOST cost-effectively?

- A. AWS Step Functions
- B. AWS Glue workflows
- C. AWS Glue Studio
- D. Amazon Managed Workflows for Apache Airflow (Amazon MWAA)

Answer: B

Explanation:

AWS Glue workflows are a cost-effective way to orchestrate complex ETL jobs that involve multiple crawlers, jobs, and triggers. AWS Glue workflows allow you to

visually monitor the progress and dependencies of your ETL tasks, and automatically handle errors and retries. AWS Glue workflows also integrate with other AWS services, such as Amazon S3, Amazon Redshift, and AWS Lambda, among others, enabling you to leverage these services for your data processing workflows. AWS Glue workflows are serverless, meaning you only pay for the resources you use, and you don't have to manage any infrastructure. AWS Step Functions, AWS Glue Studio, and Amazon MWAA are also possible options for orchestrating ETL pipelines, but they have some drawbacks compared to AWS Glue workflows. AWS Step Functions is a serverless function orchestrator that can handle different types of data processing, such as real-time, batch, and stream processing. However, AWS Step Functions requires you to write code to define your state machines, which can be complex and error-prone. AWS Step Functions also charges you for every state transition, which can add up quickly for large-scale ETL pipelines.

AWS Glue Studio is a graphical interface that allows you to create and run AWS Glue ETL jobs without writing code. AWS Glue Studio simplifies the process of building, debugging, and monitoring your ETL jobs, and provides a range of pre-built transformations and connectors. However, AWS Glue Studio does not support workflows, meaning you cannot orchestrate multiple ETL jobs or crawlers with dependencies and triggers. AWS Glue Studio also does not support streaming data sources or targets, which limits its use cases for real-time data processing.

Amazon MWAA is a fully managed service that makes it easy to run open-source versions of Apache Airflow on AWS and build workflows to run your ETL jobs and data pipelines. Amazon MWAA provides a familiar and flexible environment for data engineers who are familiar with Apache Airflow, and integrates with a range of AWS services such as Amazon EMR, AWS Glue, and AWS Step Functions. However, Amazon MWAA is not serverless, meaning you have to provision and pay for the resources you need, regardless of your usage. Amazon MWAA also requires you to write code to define your DAGs, which can be challenging and time-consuming for complex ETL pipelines. References:

? AWS Glue Workflows

? AWS Step Functions

? AWS Glue Studio

? Amazon MWAA

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 8

A company receives .csv files that contain physical address data. The data is in columns that have the following names: Door_No, Street_Name, City, and Zip_Code. The company wants to create a single column to store these values in the following format:

```
{
  "Door_No": "24",
  "Street_Name": "AAA street",
  "City": "BBB",
  "Zip_Code": "111111"
}
```

Which solution will meet this requirement with the LEAST coding effort?

- A. Use AWS Glue DataBrew to read the file
- B. Use the NEST TO ARRAY transformation to create the new column.
- C. Use AWS Glue DataBrew to read the file
- D. Use the NEST TO MAP transformation to create the new column.
- E. Use AWS Glue DataBrew to read the file
- F. Use the PIVOT transformation to create the new column.
- G. Write a Lambda function in Python to read the file
- H. Use the Python data dictionary type to create the new column.

Answer: B

Explanation:

The NEST TO MAP transformation allows you to combine multiple columns into a single column that contains a JSON object with key-value pairs. This is the easiest way to achieve the desired format for the physical address data, as you can simply select the columns to nest and specify the keys for each column. The NEST TO ARRAY transformation creates a single column that contains an array of values, which is not the same as the JSON object format. The PIVOT transformation reshapes the data by creating new columns from unique values in a selected column, which is not applicable for this use case. Writing a Lambda function in Python requires more coding effort than using AWS Glue DataBrew, which provides a visual and interactive interface for data transformations.

References:

? 7 most common data preparation transformations in AWS Glue DataBrew (Section: Nesting and unnesting columns)

? NEST TO MAP - AWS Glue DataBrew (Section: Syntax)

NEW QUESTION 9

A data engineer is building a data pipeline on AWS by using AWS Glue extract, transform, and load (ETL) jobs. The data engineer needs to process data from Amazon RDS and MongoDB, perform transformations, and load the transformed data into Amazon Redshift for analytics. The data updates must occur every hour. Which combination of tasks will meet these requirements with the LEAST operational overhead? (Choose two.)

- A. Configure AWS Glue triggers to run the ETL jobs even/ hour.
- B. Use AWS Glue DataBrew to clean and prepare the data for analytics.
- C. Use AWS Lambda functions to schedule and run the ETL jobs even/ hour.
- D. Use AWS Glue connections to establish connectivity between the data sources and Amazon Redshift.
- E. Use the Redshift Data API to load transformed data into Amazon Redshift.

Answer: AD

Explanation:

The correct answer is to configure AWS Glue triggers to run the ETL jobs every hour and use AWS Glue connections to establish connectivity between the data sources and Amazon Redshift. AWS Glue triggers are a way to schedule and orchestrate ETL jobs with the least operational overhead. AWS Glue connections are a way to securely connect to data sources and targets using JDBC or MongoDB drivers. AWS Glue DataBrew is a visual data preparation tool that does not support MongoDB as a data source. AWS Lambda functions are a serverless option to schedule and run ETL jobs, but they have a limit of 15 minutes for execution time, which may not be enough for complex transformations. The Redshift Data API is a way to run SQL commands on Amazon Redshift clusters without needing a persistent connection, but it does not support loading data from AWS Glue ETL jobs. References:

? AWS Glue triggers

? AWS Glue connections
? AWS Glue DataBrew
? [AWS Lambda functions]
? [Redshift Data API]

NEW QUESTION 10

A company needs to set up a data catalog and metadata management for data sources that run in the AWS Cloud. The company will use the data catalog to maintain the metadata of all the objects that are in a set of data stores. The data stores include structured sources such as Amazon RDS and Amazon Redshift. The data stores also include semistructured sources such as JSON files and .xml files that are stored in Amazon S3. The company needs a solution that will update the data catalog on a regular basis. The solution also must detect changes to the source metadata. Which solution will meet these requirements with the LEAST operational overhead?

- A. Use Amazon Aurora as the data catalog
- B. Create AWS Lambda functions that will connect to the data catalog
- C. Configure the Lambda functions to gather the metadata information from multiple sources and to update the Aurora data catalog
- D. Schedule the Lambda functions to run periodically.
- E. Use the AWS Glue Data Catalog as the central metadata repository
- F. Use AWS Glue crawlers to connect to multiple data stores and to update the Data Catalog with metadata change
- G. Schedule the crawlers to run periodically to update the metadata catalog.
- H. Use Amazon DynamoDB as the data catalog
- I. Create AWS Lambda functions that will connect to the data catalog
- J. Configure the Lambda functions to gather the metadata information from multiple sources and to update the DynamoDB data catalog
- K. Schedule the Lambda functions to run periodically.
- L. Use the AWS Glue Data Catalog as the central metadata repository
- M. Extract the schema for Amazon RDS and Amazon Redshift sources, and build the Data Catalog
- N. Use AWS Glue crawlers for data that is in Amazon S3 to infer the schema and to automatically update the Data Catalog.

Answer: B

Explanation:

This solution will meet the requirements with the least operational overhead because it uses the AWS Glue Data Catalog as the central metadata repository for data sources that run in the AWS Cloud. The AWS Glue Data Catalog is a fully managed service that provides a unified view of your data assets across AWS and on-premises data sources. It stores the metadata of your data in tables, partitions, and columns, and enables you to access and query your data using various AWS services, such as Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum. You can use AWS Glue crawlers to connect to multiple data stores, such as Amazon RDS, Amazon Redshift, and Amazon S3, and to update the Data Catalog with metadata changes. AWS Glue crawlers can automatically discover the schema and partition structure of your data, and create or update the corresponding tables in the Data Catalog. You can schedule the crawlers to run periodically to update the metadata catalog, and configure them to detect changes to the source metadata, such as new columns, tables, or partitions¹².

The other options are not optimal for the following reasons:

? A. Use Amazon Aurora as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the Aurora data catalog. Schedule the Lambda functions to run periodically. This option is not recommended, as it would require more operational overhead to create and manage an Amazon Aurora database as the data catalog, and to write and maintain AWS Lambda functions to gather and update the metadata information from multiple sources. Moreover, this option would not leverage the benefits of the AWS Glue Data Catalog, such as data cataloging, data transformation, and data governance.

? C. Use Amazon DynamoDB as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the DynamoDB data catalog. Schedule the Lambda functions to run periodically. This option is also not recommended, as it would require more operational overhead to create and manage an Amazon DynamoDB table as the data catalog, and to write and maintain AWS Lambda functions to gather and update the metadata information from multiple sources. Moreover, this option would not leverage the benefits of the AWS Glue Data Catalog, such as data cataloging, data transformation, and data governance.

? D. Use the AWS Glue Data Catalog as the central metadata repository. Extract the schema for Amazon RDS and Amazon Redshift sources, and build the Data Catalog. Use AWS Glue crawlers for data that is in Amazon S3 to infer the schema and to automatically update the Data Catalog. This option is not optimal, as it would require more manual effort to extract the schema for Amazon RDS and Amazon Redshift sources, and to build the Data Catalog. This option would not take advantage of the AWS Glue crawlers' ability to automatically discover the schema and partition structure of your data from various data sources, and to create or update the corresponding tables in the Data Catalog.

References:

- ? 1: AWS Glue Data Catalog
- ? 2: AWS Glue Crawlers
- ? : Amazon Aurora
- ? : AWS Lambda
- ? : Amazon DynamoDB

NEW QUESTION 10

A security company stores IoT data that is in JSON format in an Amazon S3 bucket. The data structure can change when the company upgrades the IoT devices. The company wants to create a data catalog that includes the IoT data. The company's analytics department will use the data catalog to index the data. Which solution will meet these requirements MOST cost-effectively?

- A. Create an AWS Glue Data Catalog
- B. Configure an AWS Glue Schema Registry
- C. Create a new AWS Glue workload to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.
- D. Create an Amazon Redshift provisioned cluster
- E. Create an Amazon Redshift Spectrum database for the analytics department to explore the data that is in Amazon S3. Create Redshift stored procedures to load the data into Amazon Redshift.
- F. Create an Amazon Athena workgroup
- G. Explore the data that is in Amazon S3 by using Apache Spark through Athena
- H. Provide the Athena workgroup schema and tables to the analytics department.
- I. Create an AWS Glue Data Catalog
- J. Configure an AWS Glue Schema Registry
- K. Create AWS Lambda user defined functions (UDFs) by using the Amazon Redshift Data API
- L. Create an AWS Step Functions job to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.

Answer: C

Explanation:

The best solution to meet the requirements of creating a data catalog that includes the IoT data, and allowing the analytics department to index the data, most cost-effectively, is to create an Amazon Athena workgroup, explore the data that is in Amazon S3 by using Apache Spark through Athena, and provide the Athena workgroup schema and tables to the analytics department.

Amazon Athena is a serverless, interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL or Python¹. Amazon Athena also supports Apache Spark, an open-source distributed processing framework that can run large-scale data analytics applications across clusters of servers². You can use Athena to run Spark code on data in Amazon S3 without having to set up, manage, or scale any infrastructure. You can also use Athena to create and manage external tables that point to your data in Amazon S3, and store them in an external data catalog, such as AWS Glue Data Catalog, Amazon Athena Data Catalog, or your own Apache Hive metastore³. You can create Athena workgroups to separate query execution and resource allocation based on different criteria, such as users, teams, or applications⁴. You can share the schemas and tables in your Athena workgroup with other users or applications, such as Amazon QuickSight, for data visualization and analysis⁵.

Using Athena and Spark to create a data catalog and explore the IoT data in Amazon S3 is the most cost-effective solution, as you pay only for the queries you run or the compute you use, and you pay nothing when the service is idle¹. You also save on the operational overhead and complexity of managing data warehouse infrastructure, as Athena and Spark are serverless and scalable. You can also benefit from the flexibility and performance of Athena and Spark, as they support various data formats, including JSON, and can handle schema changes and complex queries efficiently.

Option A is not the best solution, as creating an AWS Glue Data Catalog, configuring an AWS Glue Schema Registry, creating a new AWS Glue workload to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless, would incur more costs and complexity than using Athena and Spark. AWS Glue Data Catalog is a persistent metadata store that contains table definitions, job definitions, and other control information to help you manage your AWS Glue components⁶. AWS Glue Schema Registry is a service that allows you to centrally store and manage the schemas of your streaming data in AWS Glue Data Catalog⁷. AWS Glue is a serverless data integration service that makes it easy to prepare, clean, enrich, and move data between data stores⁸. Amazon Redshift Serverless is a feature of Amazon Redshift, a fully managed data warehouse service, that allows you to run and scale analytics without having to manage data warehouse infrastructure⁹. While these services are powerful and useful for many data engineering scenarios, they are not necessary or cost-effective for creating a data catalog and indexing the IoT data in Amazon S3. AWS Glue Data Catalog and Schema Registry charge you based on the number of objects stored and the number of requests made^{6,7}. AWS Glue charges you based on the compute time and the data processed by your ETL jobs⁸. Amazon Redshift Serverless charges you based on the amount of data scanned by your queries and the compute time used by your workloads⁹. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using AWS Glue and Amazon Redshift Serverless would introduce additional latency and complexity, as you would have to ingest the data from Amazon S3 to Amazon Redshift Serverless, and then query it from there, instead of querying it directly from Amazon S3 using Athena and Spark.

Option B is not the best solution, as creating an Amazon Redshift provisioned cluster, creating an Amazon Redshift Spectrum database for the analytics department to explore the data that is in Amazon S3, and creating Redshift stored procedures to load the data into Amazon Redshift, would incur more costs and complexity than using Athena and Spark. Amazon Redshift provisioned clusters are clusters that you create and manage by specifying the number and type of nodes, and the amount of storage and compute capacity¹⁰. Amazon Redshift Spectrum is a feature of Amazon Redshift that allows you to query and join data across your data warehouse and your data lake using standard SQL¹¹. Redshift stored procedures are SQL statements that you can define and store in Amazon Redshift, and then call them by using the CALL command¹². While these features are powerful and useful for many data warehousing scenarios, they are not necessary or cost-effective for creating a data catalog and indexing the IoT data in Amazon S3. Amazon Redshift provisioned clusters charge you based on the node type, the number of nodes, and the duration of the cluster¹⁰. Amazon Redshift Spectrum charges you based on the amount of data scanned by your queries¹¹. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using Amazon Redshift provisioned clusters and Spectrum would introduce additional latency and complexity, as you would have to provision and manage the cluster, create an external schema and database for the data in Amazon S3, and load the data into the cluster using stored procedures, instead of querying it directly from Amazon S3 using Athena and Spark. Option D is not the best solution, as creating an AWS Glue Data Catalog, configuring an AWS Glue Schema Registry, creating AWS Lambda user defined functions (UDFs) by using the Amazon Redshift Data API, and creating an AWS Step Functions job to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless, would incur more costs and complexity than using Athena and Spark. AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers¹³. AWS Lambda UDFs are Lambda functions that you can invoke from within an Amazon Redshift query. Amazon Redshift Data API is a service that allows you to run SQL statements on Amazon Redshift clusters using HTTP requests, without needing a persistent connection. AWS Step Functions is a service that lets you coordinate multiple AWS services into serverless workflows. While these services are powerful and useful for many data engineering scenarios, they are not necessary or cost-effective for creating a data catalog and indexing the IoT data in Amazon S3. AWS Glue Data Catalog and Schema Registry charge you based on the number of objects stored and the number of requests made^{6,7}. AWS Lambda charges you based on the number of requests and the duration of your functions¹³. Amazon Redshift Serverless charges you based on the amount of data scanned by your queries and the compute time used by your workloads⁹. AWS Step Functions charges you based on the number of state transitions in your workflows. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using AWS Glue, AWS Lambda, Amazon Redshift Data API, and AWS Step Functions would introduce additional latency and complexity, as you would have to create and invoke Lambda functions to ingest the data from Amazon S3 to Amazon Redshift Serverless using the Data API, and coordinate the ingestion process using Step Functions, instead of querying it directly from Amazon S3 using Athena and Spark. References:

? What is Amazon Athena?

? Apache Spark on Amazon Athena

? Creating tables, updating the schema, and adding new partitions in the Data Catalog from AWS Glue ETL jobs

? Managing Athena workgroups

? Using Amazon QuickSight to visualize data in Amazon Athena

? AWS Glue Data Catalog

? AWS Glue Schema Registry

? What is AWS Glue?

? Amazon Redshift Serverless

? Amazon Redshift provisioned clusters

? Querying external data using Amazon Redshift Spectrum

? Using stored procedures in Amazon Redshift

? What is AWS Lambda?

? [Creating and using AWS Lambda UDFs]

? [Using the Amazon Redshift Data API]

? [What is AWS Step Functions?]

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 15

During a security review, a company identified a vulnerability in an AWS Glue job. The company discovered that credentials to access an Amazon Redshift cluster were hard coded in the job script.

A data engineer must remediate the security vulnerability in the AWS Glue job. The solution must securely store the credentials.

Which combination of steps should the data engineer take to meet these requirements? (Choose two.)

- A. Store the credentials in the AWS Glue job parameters.
- B. Store the credentials in a configuration file that is in an Amazon S3 bucket.
- C. Access the credentials from a configuration file that is in an Amazon S3 bucket by using the AWS Glue job.
- D. Store the credentials in AWS Secrets Manager.
- E. Grant the AWS Glue job 1AM role access to the stored credentials.

Answer: DE

Explanation:

AWS Secrets Manager is a service that allows you to securely store and manage secrets, such as database credentials, API keys, passwords, etc. You can use Secrets Manager to encrypt, rotate, and audit your secrets, as well as to control access to them using fine-grained policies. AWS Glue is a fully managed service that provides a serverless data integration platform for data preparation, data cataloging, and data loading. AWS Glue jobs allow you to transform and load data from various sources into various targets, using either a graphical interface (AWS Glue Studio) or a code-based interface (AWS Glue console or AWS Glue API). Storing the credentials in AWS Secrets Manager and granting the AWS Glue job 1AM role access to the stored credentials will meet the requirements, as it will remediate the security vulnerability in the AWS Glue job and securely store the credentials. By using AWS Secrets Manager, you can avoid hard coding the credentials in the job script, which is a bad practice that exposes the credentials to unauthorized access or leakage. Instead, you can store the credentials as a secret in Secrets Manager and reference the secret name or ARN in the job script. You can also use Secrets Manager to encrypt the credentials using AWS Key Management Service (AWS KMS), rotate the credentials automatically or on demand, and monitor the access to the credentials using AWS CloudTrail. By granting the AWS Glue job 1AM role access to the stored credentials, you can use the principle of least privilege to ensure that only the AWS Glue job can retrieve the credentials from Secrets Manager. You can also use resource-based or tag-based policies to further restrict the access to the credentials.

The other options are not as secure as storing the credentials in AWS Secrets Manager and granting the AWS Glue job 1AM role access to the stored credentials. Storing the credentials in the AWS Glue job parameters will not remediate the security vulnerability, as the job parameters are still visible in the AWS Glue console and API. Storing the credentials in a configuration file that is in an Amazon S3 bucket and accessing the credentials from the configuration file by using the AWS Glue job will not be as secure as using Secrets Manager, as the configuration file may not be encrypted or rotated, and the access to the file may not be audited or controlled. References:

? AWS Secrets Manager

? AWS Glue

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 6: Data Integration and Transformation, Section 6.1: AWS Glue

NEW QUESTION 19

A data engineer must orchestrate a series of Amazon Athena queries that will run every day. Each query can run for more than 15 minutes.

Which combination of steps will meet these requirements MOST cost-effectively? (Choose two.)

- A. Use an AWS Lambda function and the Athena Boto3 client start_query_execution API call to invoke the Athena queries programmatically.
- B. Create an AWS Step Functions workflow and add two state
- C. Add the first state before the Lambda function
- D. Configure the second state as a Wait state to periodically check whether the Athena query has finished using the Athena Boto3 get_query_execution API call
- E. Configure the workflow to invoke the next query when the current query has finished running.
- F. Use an AWS Glue Python shell job and the Athena Boto3 client start_query_execution API call to invoke the Athena queries programmatically.
- G. Use an AWS Glue Python shell script to run a sleep timer that checks every 5 minutes to determine whether the current Athena query has finished running successfully
- H. Configure the Python shell script to invoke the next query when the current query has finished running.
- I. Use Amazon Managed Workflows for Apache Airflow (Amazon MWAA) to orchestrate the Athena queries in AWS Batch.

Answer: AB

Explanation:

Option A and B are the correct answers because they meet the requirements most cost-effectively. Using an AWS Lambda function and the Athena Boto3 client start_query_execution API call to invoke the Athena queries programmatically is a simple and scalable way to orchestrate the queries. Creating an AWS Step Functions workflow and adding two states to check the query status and invoke the next query is a reliable and efficient way to handle the long-running queries. Option C is incorrect because using an AWS Glue Python shell job to invoke the Athena queries programmatically is more expensive than using a Lambda function, as it requires provisioning and running a Glue job for each query.

Option D is incorrect because using an AWS Glue Python shell script to run a sleep timer that checks every 5 minutes to determine whether the current Athena query has finished running successfully is not a cost-effective or reliable way to orchestrate the queries, as it wastes resources and time.

Option E is incorrect because using Amazon Managed Workflows for Apache Airflow (Amazon MWAA) to orchestrate the Athena queries in AWS Batch is an overkill solution that introduces unnecessary complexity and cost, as it requires setting up and managing an Airflow environment and an AWS Batch compute environment.

References:

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 5: Data Orchestration, Section 5.2: AWS Lambda, Section 5.3: AWS Step Functions, Pages 125-135

? Building Batch Data Analytics Solutions on AWS, Module 5: Data Orchestration, Lesson 5.1: AWS Lambda, Lesson 5.2: AWS Step Functions, Pages 1-15

? AWS Documentation Overview, AWS Lambda Developer Guide, Working with AWS Lambda Functions, Configuring Function Triggers, Using AWS Lambda with Amazon Athena, Pages 1-4

? AWS Documentation Overview, AWS Step Functions Developer Guide, Getting Started, Tutorial: Create a Hello World Workflow, Pages 1-8

NEW QUESTION 20

A company uses an on-premises Microsoft SQL Server database to store financial transaction data. The company migrates the transaction data from the on-premises database to AWS at the end of each month. The company has noticed that the cost to migrate data from the on-premises database to an Amazon RDS for SQL Server database has increased recently.

The company requires a cost-effective solution to migrate the data to AWS. The solution must cause minimal downtime for the applications that access the database.

Which AWS service should the company use to meet these requirements?

- A. AWS Lambda
- B. AWS Database Migration Service (AWS DMS)
- C. AWS Direct Connect
- D. AWS DataSync

Answer: B

Explanation:

AWS Database Migration Service (AWS DMS) is a cloud service that makes it possible to migrate relational databases, data warehouses, NoSQL databases, and other types of data stores to AWS quickly, securely, and with minimal downtime and zero data loss¹. AWS DMS supports migration between 20-plus database and analytics engines, such as Microsoft SQL Server to Amazon RDS for SQL Server². AWS DMS takes over many of the difficult or tedious tasks involved in a migration project, such as capacity analysis, hardware and software procurement, installation and administration, testing and debugging, and ongoing replication and monitoring¹. AWS DMS is a cost-effective solution, as you only pay for the compute resources and additional log storage used during the migration process². AWS DMS is the best solution for the company to migrate the financial transaction data from the on-premises Microsoft SQL Server database to AWS, as it meets

the requirements of minimal downtime, zero data loss, and low cost.

Option A is not the best solution, as AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers, but it does not provide any built-in features for database migration. You would have to write your own code to extract, transform, and load the data from the source to the target, which would increase the operational overhead and complexity.

Option C is not the best solution, as AWS Direct Connect is a service that establishes a dedicated network connection from your premises to AWS, but it does not provide any built-in features for database migration. You would still need to use another service or tool to perform the actual data transfer, which would increase the cost and complexity.

Option D is not the best solution, as AWS DataSync is a service that makes it easy to transfer data between on-premises storage systems and AWS storage services, such as Amazon S3, Amazon EFS, and Amazon FSx for Windows File Server, but it does not support Amazon RDS for SQL Server as a target. You would have to use another service or tool to migrate the data from Amazon S3 to Amazon RDS for SQL Server, which would increase the latency and complexity.

References:

? Database Migration - AWS Database Migration Service - AWS

? What is AWS Database Migration Service?

? AWS Database Migration Service Documentation

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 24

A media company wants to improve a system that recommends media content to customer based on user behavior and preferences. To improve the recommendation system, the company needs to incorporate insights from third-party datasets into the company's existing analytics platform.

The company wants to minimize the effort and time required to incorporate third-party datasets.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Use API calls to access and integrate third-party datasets from AWS Data Exchange.
- B. Use API calls to access and integrate third-party datasets from AWS
- C. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from AWS CodeCommit repositories.
- D. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from Amazon Elastic Container Registry (Amazon ECR).

Answer: A

Explanation:

AWS Data Exchange is a service that makes it easy to find, subscribe to, and use third-party data in the cloud. It provides a secure and reliable way to access and integrate data from various sources, such as data providers, public datasets, or AWS services. Using AWS Data Exchange, you can browse and subscribe to data products that suit your needs, and then use API calls or the AWS Management Console to export the data to Amazon S3, where you can use it with your existing analytics platform. This solution minimizes the effort and time required to incorporate third-party datasets, as you do not need to set up and manage data pipelines, storage, or access controls. You also benefit from the data quality and freshness provided by the data providers, who can update their data products as frequently as needed¹².

The other options are not optimal for the following reasons:

? B. Use API calls to access and integrate third-party datasets from AWS. This option is vague and does not specify which AWS service or feature is used to access and integrate third-party datasets. AWS offers a variety of services and features that can help with data ingestion, processing, and analysis, but not all of them are suitable for the given scenario. For example, AWS Glue is a serverless data integration service that can help you discover, prepare, and combine data from various sources, but it requires you to create and run data extraction, transformation, and loading (ETL) jobs, which can add operational overhead³.

? C. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from AWS CodeCommit repositories. This option is not feasible, as AWS CodeCommit is a source control service that hosts secure Git-based repositories, not a data source that can be accessed by Amazon Kinesis Data Streams. Amazon Kinesis Data Streams is a service that enables you to capture, process, and analyze data streams in real time, such as clickstream data, application logs, or IoT telemetry. It does not support accessing and integrating data from AWS CodeCommit repositories, which are meant for storing and managing code, not data .

? D. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from Amazon Elastic Container Registry (Amazon ECR). This option is also not feasible, as Amazon ECR is a fully managed container registry service that stores, manages, and deploys container images, not a data source that can be accessed by Amazon Kinesis Data Streams. Amazon Kinesis Data Streams does not support accessing and integrating data from Amazon ECR, which is meant for storing and managing container images, not data .

References:

? 1: AWS Data Exchange User Guide

? 2: AWS Data Exchange FAQs

? 3: AWS Glue Developer Guide

? : AWS CodeCommit User Guide

? : Amazon Kinesis Data Streams Developer Guide

? : Amazon Elastic Container Registry User Guide

? : Build a Continuous Delivery Pipeline for Your Container Images with Amazon ECR as Source

NEW QUESTION 25

An airline company is collecting metrics about flight activities for analytics. The company is conducting a proof of concept (POC) test to show how analytics can provide insights that the company can use to increase on-time departures.

The POC test uses objects in Amazon S3 that contain the metrics in .csv format. The POC test uses Amazon Athena to query the data. The data is partitioned in the S3 bucket by date.

As the amount of data increases, the company wants to optimize the storage solution to improve query performance.

Which combination of solutions will meet these requirements? (Choose two.)

- A. Add a randomized string to the beginning of the keys in Amazon S3 to get more throughput across partitions.
- B. Use an S3 bucket that is in the same account that uses Athena to query the data.
- C. Use an S3 bucket that is in the same AWS Region where the company runs Athena queries.
- D. Preprocess the .csv data to JSON format by fetching only the document keys that the query requires.
- E. Preprocess the .csv data to Apache Parquet format by fetching only the data blocks that are needed for predicates.

Answer: CE

Explanation:

Using an S3 bucket that is in the same AWS Region where the company runs Athena queries can improve query performance by reducing data transfer latency and costs. Preprocessing the .csv data to Apache Parquet format can also improve query performance by enabling columnar storage, compression, and partitioning, which can reduce the amount of data scanned and fetched by the query. These solutions can optimize the storage solution for the POC test without requiring much effort or changes to the existing data pipeline. The other solutions are not optimal or relevant for this requirement. Adding a randomized string to the beginning of the keys in Amazon S3 can improve the throughput across partitions, but it can also make the data harder to query and manage. Using an S3

bucket that is in the same account that uses Athena to query the data does not have any significant impact on query performance, as long as the proper permissions are granted. Preprocessing the .csv data to JSON format does not offer any benefits over the .csv format, as both are row-based and verbose formats that require more data scanning and fetching than columnar formats like Parquet. References:

? Best Practices When Using Athena with AWS Glue

? Optimizing Amazon S3 Performance

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 29

A company uses an Amazon Redshift cluster that runs on RA3 nodes. The company wants to scale read and write capacity to meet demand. A data engineer needs to identify a solution that will turn on concurrency scaling.

Which solution will meet this requirement?

- A. Turn on concurrency scaling in workload management (WLM) for Redshift Serverless workgroups.
- B. Turn on concurrency scaling at the workload management (WLM) queue level in the Redshift cluster.
- C. Turn on concurrency scaling in the settings during the creation of a new Redshift cluster.
- D. Turn on concurrency scaling for the daily usage quota for the Redshift cluster.

Answer: B

Explanation:

Concurrency scaling is a feature that allows you to support thousands of concurrent users and queries, with consistently fast query performance. When you turn on concurrency scaling, Amazon Redshift automatically adds query processing power in seconds to process queries without any delays. You can manage which queries are sent to the concurrency-scaling cluster by configuring WLM queues. To turn on concurrency scaling for a queue, set the Concurrency Scaling mode value to auto. The other options are either incorrect or irrelevant, as they do not enable concurrency scaling for the existing Redshift cluster on RA3 nodes.

References:

? Working with concurrency scaling - Amazon Redshift

? Amazon Redshift Concurrency Scaling - Amazon Web Services

? Configuring concurrency scaling queues - Amazon Redshift

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 6, page 163)

NEW QUESTION 30

A company stores daily records of the financial performance of investment portfolios in .csv format in an Amazon S3 bucket. A data engineer uses AWS Glue crawlers to crawl the S3 data.

The data engineer must make the S3 data accessible daily in the AWS Glue Data Catalog. Which solution will meet these requirements?

- A. Create an IAM role that includes the AmazonS3FullAccess policy
- B. Associate the role with the crawler
- C. Specify the S3 bucket path of the source data as the crawler's data store
- D. Create a daily schedule to run the crawler
- E. Configure the output destination to a new path in the existing S3 bucket.
- F. Create an IAM role that includes the AWSGlueServiceRole policy
- G. Associate the role with the crawler
- H. Specify the S3 bucket path of the source data as the crawler's data store
- I. Create a daily schedule to run the crawler
- J. Specify a database name for the output.
- K. Create an IAM role that includes the AmazonS3FullAccess policy
- L. Associate the role with the crawler
- M. Specify the S3 bucket path of the source data as the crawler's data store
- N. Allocate data processing units (DPU) to run the crawler every day
- O. Specify a database name for the output.
- P. Create an IAM role that includes the AWSGlueServiceRole policy
- Q. Associate the role with the crawler
- R. Specify the S3 bucket path of the source data as the crawler's data store
- S. Allocate data processing units (DPU) to run the crawler every day
- T. Configure the output destination to a new path in the existing S3 bucket.

Answer: B

Explanation:

To make the S3 data accessible daily in the AWS Glue Data Catalog, the data engineer needs to create a crawler that can crawl the S3 data and write the metadata to the Data Catalog. The crawler also needs to run on a daily schedule to keep the Data Catalog updated with the latest data. Therefore, the solution must include the following steps:

? Create an IAM role that has the necessary permissions to access the S3 data and

the Data Catalog. The AWSGlueServiceRole policy is a managed policy that grants these permissions¹.

? Associate the role with the crawler.

? Specify the S3 bucket path of the source data as the crawler's data store. The crawler will scan the data and infer the schema and format².

? Create a daily schedule to run the crawler. The crawler will run at the specified time every day and update the Data Catalog with any changes in the data³.

? Specify a database name for the output. The crawler will create or update a table in the Data Catalog under the specified database. The table will contain the metadata about the data in the S3 bucket, such as the location, schema, and classification.

Option B is the only solution that includes all these steps. Therefore, option B is the correct answer.

Option A is incorrect because it configures the output destination to a new path in the existing S3 bucket. This is unnecessary and may cause confusion, as the crawler does not write any data to the S3 bucket, only metadata to the Data Catalog.

Option C is incorrect because it allocates data processing units (DPUs) to run the crawler every day. This is also unnecessary, as DPUs are only used for AWS Glue ETL jobs, not crawlers.

Option D is incorrect because it combines the errors of option A and C. It configures the output destination to a new path in the existing S3 bucket and allocates DPUs to run the crawler every day, both of which are irrelevant for the crawler.

References:

? 1: AWS managed (predefined) policies for AWS Glue - AWS Glue

? 2: Data Catalog and crawlers in AWS Glue - AWS Glue

? 3: Scheduling an AWS Glue crawler - AWS Glue

? [4]: Parameters set on Data Catalog tables by crawler - AWS Glue

? [5]: AWS Glue pricing - Amazon Web Services (AWS)

NEW QUESTION 31

A company is developing an application that runs on Amazon EC2 instances. Currently, the data that the application generates is temporary. However, the company needs to persist the data, even if the EC2 instances are terminated.

A data engineer must launch new EC2 instances from an Amazon Machine Image (AMI) and configure the instances to preserve the data.

Which solution will meet this requirement?

- A. Launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume that contains the application data
- B. Apply the default settings to the EC2 instances.
- C. Launch new EC2 instances by using an AMI that is backed by a root Amazon Elastic Block Store (Amazon EBS) volume that contains the application data
- D. Apply the default settings to the EC2 instances.
- E. Launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume
- F. Attach an Amazon Elastic Block Store (Amazon EBS) volume to contain the application data
- G. Apply the default settings to the EC2 instances.
- H. Launch new EC2 instances by using an AMI that is backed by an Amazon Elastic Block Store (Amazon EBS) volume
- I. Attach an additional EC2 instance store volume to contain the application data
- J. Apply the default settings to the EC2 instances.

Answer: C

Explanation:

Amazon EC2 instances can use two types of storage volumes: instance store volumes and Amazon EBS volumes. Instance store volumes are ephemeral, meaning they are only attached to the instance for the duration of its life cycle. If the instance is stopped, terminated, or fails, the data on the instance store volume is lost. Amazon EBS volumes are persistent, meaning they can be detached from the instance and attached to another instance, and the data on the volume is preserved. To meet the requirement of persisting the data even if the EC2 instances are terminated, the data engineer must use Amazon EBS volumes to store the application data. The solution is to launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume, which is the default option for most AMIs. Then, the data engineer must attach an Amazon EBS volume to each instance and configure the application to write the data to the EBS volume. This way, the data will be saved on the EBS volume and can be accessed by another instance if needed. The data engineer can apply the default settings to the EC2 instances, as there is no need to modify the instance type, security group, or IAM role for this solution. The other options are either not feasible or not optimal. Launching new EC2 instances by using an AMI that is backed by an EC2 instance store volume that contains the application data (option A) or by using an AMI that is backed by a root Amazon EBS volume that contains the application data (option B) would not work, as the data on the AMI would be outdated and overwritten by the new instances. Attaching an additional EC2 instance store volume to contain the application data (option D) would not work, as the data on the instance store volume would be lost if the instance is terminated. References:

? Amazon EC2 Instance Store

? Amazon EBS Volumes

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 2: Data Store Management, Section 2.1: Amazon EC2

NEW QUESTION 34

A company uses an Amazon Redshift provisioned cluster as its database. The Redshift cluster has five reserved ra3.4xlarge nodes and uses key distribution.

A data engineer notices that one of the nodes frequently has a CPU load over 90%. SQL Queries that run on the node are queued. The other four nodes usually have a CPU load under 15% during daily operations.

The data engineer wants to maintain the current number of compute nodes. The data engineer also wants to balance the load more evenly across all five compute nodes.

Which solution will meet these requirements?

- A. Change the sort key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement.
- B. Change the distribution key to the table column that has the largest dimension.
- C. Upgrade the reserved node from ra3.4xlarge to ra3.16xlarge.
- D. Change the primary key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement.

Answer: B

Explanation:

Changing the distribution key to the table column that has the largest dimension will help to balance the load more evenly across all five compute nodes. The distribution key determines how the rows of a table are distributed among the slices of the cluster. If the distribution key is not chosen wisely, it can cause data skew, meaning some slices will have more data than others, resulting in uneven CPU load and query performance. By choosing the table column that has the largest dimension, meaning the column that has the most distinct values, as the distribution key, the data engineer can ensure that the rows are distributed more uniformly across the slices, reducing data skew and improving query performance.

The other options are not solutions that will meet the requirements. Option A, changing the sort key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement, will not affect the data distribution or the CPU load. The sort key determines the order in which the rows of a table are stored on disk, which can improve the performance of range-restricted queries, but not the load balancing. Option C, upgrading the reserved node from ra3.4xlarge to ra3.16xlarge, will not maintain the current number of compute nodes, as it will increase the cost and the capacity of the cluster. Option D, changing the primary key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement, will not affect the data distribution or the CPU load either.

The primary key is a constraint that enforces the uniqueness of the rows in a table, but it does not influence the data layout or the query optimization. References:

? Choosing a data distribution style

? Choosing a data sort key

? Working with primary keys

NEW QUESTION 39

A company is migrating a legacy application to an Amazon S3 based data lake. A data engineer reviewed data that is associated with the legacy application. The data engineer found that the legacy data contained some duplicate information.

The data engineer must identify and remove duplicate information from the legacy application data.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Write a custom extract, transform, and load (ETL) job in Python
- B. Use the `DataFrame.drop_duplicates()` function by importing the Pandas library to perform data deduplication.
- C. Write an AWS Glue extract, transform, and load (ETL) job
- D. Use the `FindMatches` machine learning (ML) transform to transform the data to perform data deduplication.
- E. Write a custom extract, transform, and load (ETL) job in Python

- F. Import the Python dedupe librar
- G. Use the dedupe library to perform data deduplication.
- H. Write an AWS Glue extract, transform, and load (ETL) jo
- I. Import the Python dedupe librar
- J. Use the dedupe library to perform data deduplication.

Answer: B

Explanation:

AWS Glue is a fully managed serverless ETL service that can handle data deduplication with minimal operational overhead. AWS Glue provides a built-in ML transform called FindMatches, which can automatically identify and group similar records in a dataset. FindMatches can also generate a primary key for each group of records and remove duplicates. FindMatches does not require any coding or prior ML experience, as it can learn from a sample of labeled data provided by the user. FindMatches can also scale to handle large datasets and optimize the cost and performance of the ETL job. References:

? AWS Glue

? FindMatches ML Transform

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 40

A healthcare company uses Amazon Kinesis Data Streams to stream real-time health data from wearable devices, hospital equipment, and patient records.

A data engineer needs to find a solution to process the streaming data. The data engineer needs to store the data in an Amazon Redshift Serverless warehouse.

The solution must support near real-time analytics of the streaming data and the previous day's data. Which solution will meet these requirements with the LEAST operational overhead?

- A. Load data into Amazon Kinesis Data Firehos
- B. Load the data into Amazon Redshift.
- C. Use the streaming ingestion feature of Amazon Redshift.
- D. Load the data into Amazon S3. Use the COPY command to load the data into Amazon Redshift.
- E. Use the Amazon Aurora zero-ETL integration with Amazon Redshift.

Answer: B

Explanation:

The streaming ingestion feature of Amazon Redshift enables you to ingest data from streaming sources, such as Amazon Kinesis Data Streams, into Amazon Redshift tables in near real-time. You can use the streaming ingestion feature to process the streaming data from the wearable devices, hospital equipment, and patient records. The streaming ingestion feature also supports incremental updates, which means you can append new data or update existing data in the Amazon Redshift tables. This way, you can store the data in an Amazon Redshift Serverless warehouse and support near real-time analytics of the streaming data and the previous day's data. This solution meets the requirements with the least operational overhead, as it does not require any additional services or components to ingest and process the streaming data. The other options are either not feasible or not optimal. Loading data into Amazon Kinesis Data Firehose and then into Amazon Redshift (option A) would introduce additional latency and cost, as well as require additional configuration and management. Loading data into Amazon S3 and then using the COPY command to load the data into Amazon Redshift (option C) would also introduce additional latency and cost, as well as require additional storage space and ETL logic. Using the Amazon Aurora zero-ETL integration with Amazon Redshift (option D) would not work, as it requires the data to be stored in Amazon Aurora first, which is not the case for the streaming data from the healthcare company. References:

? Using streaming ingestion with Amazon Redshift

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 3: Data Ingestion and Transformation, Section 3.5: Amazon Redshift Streaming Ingestion

NEW QUESTION 43

A company has five offices in different AWS Regions. Each office has its own human resources (HR) department that uses a unique IAM role. The company stores employee records in a data lake that is based on Amazon S3 storage.

A data engineering team needs to limit access to the records. Each HR department should be able to access records for only employees who are within the HR department's Region.

Which combination of steps should the data engineering team take to meet this requirement with the LEAST operational overhead? (Choose two.)

- A. Use data filters for each Region to register the S3 paths as data locations.
- B. Register the S3 path as an AWS Lake Formation location.
- C. Modify the IAM roles of the HR departments to add a data filter for each department's Region.
- D. Enable fine-grained access control in AWS Lake Formatio
- E. Add a data filter for each Region.
- F. Create a separate S3 bucket for each Regio
- G. Configure an IAM policy to allow S3 acces
- H. Restrict access based on Region.

Answer: BD

Explanation:

AWS Lake Formation is a service that helps you build, secure, and manage data lakes on Amazon S3. You can use AWS Lake Formation to register the S3 path as a data lake location, and enable fine-grained access control to limit access to the records based on the HR department's Region. You can use data filters to specify which S3 prefixes or partitions each HR department can access, and grant permissions to the IAM roles of the HR departments accordingly. This solution will meet the requirement with the least operational overhead, as it simplifies the data lake management and security, and leverages the existing IAM roles of the HR departments¹².

The other options are not optimal for the following reasons:

? A. Use data filters for each Region to register the S3 paths as data locations. This option is not possible, as data filters are not used to register S3 paths as data locations, but to grant permissions to access specific S3 prefixes or partitions within a data location. Moreover, this option does not specify how to limit access to the records based on the HR department's Region.

? C. Modify the IAM roles of the HR departments to add a data filter for each department's Region. This option is not possible, as data filters are not added to IAM roles, but to permissions granted by AWS Lake Formation. Moreover, this option does not specify how to register the S3 path as a data lake location, or how to enable fine-grained access control in AWS Lake Formation.

? E. Create a separate S3 bucket for each Region. Configure an IAM policy to allow S3 access. Restrict access based on Region. This option is not recommended, as it would require more operational overhead to create and manage multiple S3 buckets, and to configure and maintain IAM policies for each HR department. Moreover, this option does not leverage the benefits of AWS Lake Formation, such as data cataloging, data transformation, and data governance.

References:

? 1: AWS Lake Formation
? 2: AWS Lake Formation Permissions
? : AWS Identity and Access Management
? : Amazon S3

NEW QUESTION 47

A company uses Amazon RDS to store transactional data. The company runs an RDS DB instance in a private subnet. A developer wrote an AWS Lambda function with default settings to insert, update, or delete data in the DB instance.

The developer needs to give the Lambda function the ability to connect to the DB instance privately without using the public internet.

Which combination of steps will meet this requirement with the LEAST operational overhead? (Choose two.)

- A. Turn on the public access setting for the DB instance.
- B. Update the security group of the DB instance to allow only Lambda function invocations on the database port.
- C. Configure the Lambda function to run in the same subnet that the DB instance uses.
- D. Attach the same security group to the Lambda function and the DB instance.
- E. Include a self-referencing rule that allows access through the database port.
- F. Update the network ACL of the private subnet to include a self-referencing rule that allows access through the database port.

Answer: CD

Explanation:

To enable the Lambda function to connect to the RDS DB instance privately without using the public internet, the best combination of steps is to configure the Lambda function to run in the same subnet that the DB instance uses, and attach the same security group to the Lambda function and the DB instance. This way, the Lambda function and the DB instance can communicate within the same private network, and the security group can allow traffic between them on the database port. This solution has the least operational overhead, as it does not require any changes to the public access setting, the network ACL, or the security group of the DB instance.

The other options are not optimal for the following reasons:

? A. Turn on the public access setting for the DB instance. This option is not recommended, as it would expose the DB instance to the public internet, which can compromise the security and privacy of the data. Moreover, this option would not enable the Lambda function to connect to the DB instance privately, as it would still require the Lambda function to use the public internet to access the DB instance.

? B. Update the security group of the DB instance to allow only Lambda function invocations on the database port. This option is not sufficient, as it would only modify the inbound rules of the security group of the DB instance, but not the outbound rules of the security group of the Lambda function. Moreover, this option would not enable the Lambda function to connect to the DB instance privately, as it would still require the Lambda function to use the public internet to access the DB instance.

? E. Update the network ACL of the private subnet to include a self-referencing rule

that allows access through the database port. This option is not necessary, as the network ACL of the private subnet already allows all traffic within the subnet by default. Moreover, this option would not enable the Lambda function to connect to the DB instance privately, as it would still require the Lambda function to use the public internet to access the DB instance.

References:

? 1: Connecting to an Amazon RDS DB instance

? 2: Configuring a Lambda function to access resources in a VPC

? 3: Working with security groups

? : Network ACLs

NEW QUESTION 48

A data engineer has a one-time task to read data from objects that are in Apache Parquet format in an Amazon S3 bucket. The data engineer needs to query only one column of the data.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Configure an AWS Lambda function to load data from the S3 bucket into a pandas dataframe- Write a SQL SELECT statement on the dataframe to query the required column.
- B. Use S3 Select to write a SQL SELECT statement to retrieve the required column from the S3 objects.
- C. Prepare an AWS Glue DataBrew project to consume the S3 objects and to query the required column.
- D. Run an AWS Glue crawler on the S3 object
- E. Use a SQL SELECT statement in Amazon Athena to query the required column.

Answer: B

Explanation:

Option B is the best solution to meet the requirements with the least operational overhead because S3 Select is a feature that allows you to retrieve only a subset of data from an S3 object by using simple SQL expressions. S3 Select works on objects stored in CSV, JSON, or Parquet format. By using S3 Select, you can avoid the need to download and process the entire S3 object, which reduces the amount of data transferred and the computation time. S3 Select is also easy to use and does not require any additional services or resources.

Option A is not a good solution because it involves writing custom code and configuring an AWS Lambda function to load data from the S3 bucket into a pandas dataframe and query the required column. This option adds complexity and latency to the data retrieval process and requires additional resources and configuration. Moreover, AWS Lambda has limitations on the execution time, memory, and concurrency, which may affect the performance and reliability of the data retrieval process.

Option C is not a good solution because it involves creating and running an AWS Glue DataBrew project to consume the S3 objects and query the required column. AWS Glue DataBrew is a visual data preparation tool that allows you to clean, normalize, and transform data without writing code. However, in this scenario, the data is already in Parquet format, which is a columnar storage format that is optimized for analytics. Therefore, there is no need to use AWS Glue DataBrew to prepare the data. Moreover, AWS Glue DataBrew adds extra time and cost to the data retrieval process and requires additional resources and configuration.

Option D is not a good solution because it involves running an AWS Glue crawler on the S3 objects and using a SQL SELECT statement in Amazon Athena to query the required column. An AWS Glue crawler is a service that can scan data sources and create metadata tables in the AWS Glue Data Catalog. The Data Catalog is a central repository that stores information about the data sources, such as schema, format, and location. Amazon Athena is a serverless interactive query service that allows you to analyze data in S3 using standard SQL. However, in this scenario, the schema and format of the data are already known and fixed, so there is no need to run a crawler to discover them. Moreover, running a crawler and using Amazon Athena adds extra time and cost to the data retrieval process and requires additional services and configuration.

References:

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

? S3 Select and Glacier Select - Amazon Simple Storage Service

- ? AWS Lambda - FAQs
- ? What Is AWS Glue DataBrew? - AWS Glue DataBrew
- ? Populating the AWS Glue Data Catalog - AWS Glue
- ? What is Amazon Athena? - Amazon Athena

NEW QUESTION 51

A data engineer needs to maintain a central metadata repository that users access through Amazon EMR and Amazon Athena queries. The repository needs to provide the schema and properties of many tables. Some of the metadata is stored in Apache Hive. The data engineer needs to import the metadata from Hive into the central metadata repository.

Which solution will meet these requirements with the LEAST development effort?

- A. Use Amazon EMR and Apache Ranger.
- B. Use a Hive metastore on an EMR cluster.
- C. Use the AWS Glue Data Catalog.
- D. Use a metastore on an Amazon RDS for MySQL DB instance.

Answer: C

Explanation:

The AWS Glue Data Catalog is an Apache Hive metastore-compatible catalog that provides a central metadata repository for various data sources and formats. You can use the AWS Glue Data Catalog as an external Hive metastore for Amazon EMR and Amazon Athena queries, and import metadata from existing Hive metastores into the Data Catalog. This solution requires the least development effort, as you can use AWS Glue crawlers to automatically discover and catalog the metadata from Hive, and use the AWS Glue console, AWS CLI, or Amazon EMR API to configure the Data Catalog as the Hive metastore. The other options are either more complex or require additional steps, such as setting up Apache Ranger for security, managing a Hive metastore on an EMR cluster or an RDS instance, or migrating the metadata manually. References:

- ? Using the AWS Glue Data Catalog as the metastore for Hive (Section: Specifying AWS Glue Data Catalog as the metastore)
- ? Metadata Management: Hive Metastore vs AWS Glue (Section: AWS Glue Data Catalog)
- ? AWS Glue Data Catalog support for Spark SQL jobs (Section: Importing metadata from an existing Hive metastore)
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 131)

NEW QUESTION 56

A data engineer needs Amazon Athena queries to finish faster. The data engineer notices that all the files the Athena queries use are currently stored in uncompressed .csv format. The data engineer also notices that users perform most queries by selecting a specific column.

Which solution will MOST speed up the Athena query performance?

- A. Change the data format from .csv to JSON format.
- B. Apply Snappy compression.
- C. Compress the .csv files by using Snappy compression.
- D. Change the data format from .csv to Apache Parquet.
- E. Apply Snappy compression.
- F. Compress the .csv files by using gzip compression.

Answer: C

Explanation:

Amazon Athena is a serverless interactive query service that allows you to analyze data in Amazon S3 using standard SQL. Athena supports various data formats, such as CSV, JSON, ORC, Avro, and Parquet. However, not all data formats are equally efficient for querying. Some data formats, such as CSV and JSON, are row-oriented, meaning that they store data as a sequence of records, each with the same fields. Row-oriented formats are suitable for loading and exporting data, but they are not optimal for analytical queries that often access only a subset of columns. Row-oriented formats also do not support compression or encoding techniques that can reduce the data size and improve the query performance.

On the other hand, some data formats, such as ORC and Parquet, are column-oriented, meaning that they store data as a collection of columns, each with a specific data type. Column-oriented formats are ideal for analytical queries that often filter, aggregate, or join data by columns. Column-oriented formats also support compression and encoding techniques that can reduce the data size and improve the query performance. For example, Parquet supports dictionary encoding, which replaces repeated values with numeric codes, and run-length encoding, which replaces consecutive identical values with a single value and a count. Parquet also supports various compression algorithms, such as Snappy, GZIP, and ZSTD, that can further reduce the data size and improve the query performance.

Therefore, changing the data format from CSV to Parquet and applying Snappy compression will most speed up the Athena query performance. Parquet is a column-oriented format that allows Athena to scan only the relevant columns and skip the rest, reducing the amount of data read from S3. Snappy is a compression algorithm that reduces the data size without compromising the query speed, as it is splittable and does not require decompression before reading. This solution will also reduce the cost of Athena queries, as Athena charges based on the amount of data scanned from S3.

The other options are not as effective as changing the data format to Parquet and applying Snappy compression. Changing the data format from CSV to JSON and applying Snappy compression will not improve the query performance significantly, as JSON is also a row-oriented format that does not support columnar access or encoding techniques. Compressing the CSV files by using Snappy compression will reduce the data size, but it will not improve the query performance significantly, as CSV is still a row-oriented format that does not support columnar access or encoding techniques. Compressing the CSV files by using gzip compression will reduce the data size, but it will degrade the query performance, as gzip is not a splittable compression algorithm and requires decompression before reading. References:

- ? Amazon Athena
- ? Choosing the Right Data Format
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 5: Data Analysis and Visualization, Section 5.1: Amazon Athena

NEW QUESTION 61

.....

THANKS FOR TRYING THE DEMO OF OUR PRODUCT

Visit Our Site to Purchase the Full Set of Actual AWS-Certified-Data-Engineer-Associate Exam Questions With Answers.

We Also Provide Practice Exam Software That Simulates Real Exam Environment And Has Many Self-Assessment Features. Order the AWS-Certified-Data-Engineer-Associate Product From:

<https://www.2passeasy.com/dumps/AWS-Certified-Data-Engineer-Associate/>

Money Back Guarantee

AWS-Certified-Data-Engineer-Associate Practice Exam Features:

- * AWS-Certified-Data-Engineer-Associate Questions and Answers Updated Frequently
- * AWS-Certified-Data-Engineer-Associate Practice Questions Verified by Expert Senior Certified Staff
- * AWS-Certified-Data-Engineer-Associate Most Realistic Questions that Guarantee you a Pass on Your FirstTry
- * AWS-Certified-Data-Engineer-Associate Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year